

# Efficient model-based clustering with coalescents: Application to multiple outcomes using medical records data

Ricardo Henao and Joseph E. Lucas

August 11, 2016

## Abstract

We present a sequential Monte Carlo sampler for coalescent based Bayesian hierarchical clustering. The model is appropriate for multivariate non-i.i.d. data and our approach offers a substantial reduction in computational cost when compared to the original sampler. We also propose a quadratic complexity approximation that in practice shows almost no loss in performance compared to its counterpart. Our formulation leads to a greedy algorithm that exhibits performance improvement over other greedy algorithms, particularly in small data sets. We incorporate the Coalescent into a hierarchical regression model that allows joint modeling of multiple correlated outcomes. The approach does not require *a priori* knowledge of either the degree or structure of the correlation and, as a byproduct, generates additional models for a subset of the composite outcomes. We demonstrate the utility of the approach by predicting multiple different types of outcomes using medical records data from a cohort of diabetic patients.

**Keywords:** coalescent, sequential Monte Carlo, greedy algorithm, electronic medical record, predictive medicine, multiple outcomes

---

Ricardo Henao and Joseph E. Lucas are Assistant Research Professors in Electrical and Computer Engineering, Duke University, Durham, NC 27710. E-mail: r.henao@duke.edu and joe@stat.duke.edu.

# 1 Introduction

Learning hierarchical structures from observed data is a common practice in many knowledge domains. Examples include phylogenies and signaling pathways in biology, language models in linguistics, etc. Agglomerative clustering is still the most popular approach to hierarchical clustering due to its efficiency, ease of implementation and a wide range of possible distance metrics. However, because it is algorithmic in nature, there is no principled way that agglomerative clustering can be used as a building block in more complex models. Bayesian priors for structure learning on the other hand, are perfectly suited to be employed in larger models. Several authors have proposed using hierarchical structure priors to model correlation in factor models (Rai and Daume III, 2009; Zhang et al., 2011; Henao et al., 2012, 2013).

There are many approaches to hierarchical structure learning already proposed in the literature, see for instance Neal (2003a); Heller and Ghahramani (2005); Teh et al. (2008); Adams et al. (2010). The work in this paper focuses on the Bayesian agglomerative clustering model proposed by Teh et al. (2008). This allows us to perform model based hierarchical clustering with continuous multivariate non i.i.d. data – by which we mean multivariate observations in which the elements of the vector are not i.i.d. . Although the authors introduce priors both for continuous and discrete data, no attention is paid to the non i.i.d. case, mainly because their work is focused in proposing different inference alternatives.

Kingman’s coalescent is a standard model from population genetics perfectly suited for hierarchical clustering since it defines a prior over binary trees (Kingman, 1982a,b). This work advances Bayesian hierarchical clustering in two ways: (i) we extend the original model to handle non i.i.d. data and (ii) we propose an efficient sequential Monte Carlo inference procedure for the model which scales quadratically rather than cubically. As a byproduct of our approach we propose as well a small correction to the

greedy algorithm of Teh et al. (2008) that shows gains particularly in small data sets. In addition, we demonstrate a novel application of the Coalescent within a hierarchical regression that gives improved predictive accuracy in the presence of multiple correlated outcomes.

There is a separate approach by Görür and Teh (2009) that also improves the cubic computational cost of Bayesian hierarchical clustering. They introduce an efficient sampler with quadratic cost that although proposed for discrete data can be easily extended to continuous data, however as we will show, our approach is still considerably faster.

The remainder of the manuscript is organized as follows, the data model and the use of coalescents as priors for hierarchical clustering are reviewed in Section 2. Our approach to inference and relationships to previous approaches are described in Section 3. Section 4 demonstrates the use of the model and algorithm in the context of hierarchical regression with multiple outcomes of unknown correlation. The appendix contains numerical results on both artificial and real data.

## 2 Coalescents for hierarchical clustering

`<sc:coalescent>` A *partition* of a set  $A$  is defined to be a collection of subsets  $\{a_j\}_{j=1}^J$  such that (1)  $a_j \subset A$  for all  $j$ , (2)  $\cup_j a_j = A$  and (3)  $a_j \cap a_{j'} = \emptyset$  for all  $j \neq j'$ . Let  $X$  be an  $n \times d$  dimensional matrix of  $n$  observations in  $d$  dimensions with rows  $\mathbf{x}_i$ . We will define a binary tree on our set of  $n$  observations; define  $\pi_0 = \{\{\mathbf{x}_1\}, \dots, \{\mathbf{x}_n\}\}$  to be a partition containing  $n$  singletons. We iteratively combine pairs of elements such that  $\pi_{i+1}$  is obtained by removing two of the elements from  $\pi_i$  and inserting their union. Thus if  $\pi_i = \{a_j\}_{j=1}^{n-i}$  then  $\pi_{i+1} = (\pi_i \setminus \{a_j, a_{j'}\}) \cup (a_j \cup a_{j'})$  for some  $j \neq j'$ . For  $a_j, a_{j'} \in \pi_i$  define  $z_{i+1}^* = a_j \cup a_{j'}$  where  $j$  and  $j'$  indicate the elements of  $\pi_i$  that were merged to

obtain  $\pi_{i+1}$ . A particular sequence of  $n$  partitions defines a binary tree. The  $n$  leaves of the tree are the elements of  $\pi_0$  and the nodes (branching points) of the tree are  $\{z_k^*\}_{k=1}^{n-1}$ .

Defining  $\mathbf{t} = [t_1 \dots t_{n-1}]$  to be a vector of branching times,  $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_{n-1}\}$  to be the ordered collection of partitions and  $\mathbf{z} = \{\mathbf{z}_1, \dots, \mathbf{z}_{n-1}\}$  to be the latent  $d$ -dimensional observations at the internal nodes we can write a Bayesian model for hierarchical clustering as

$$\begin{aligned} \mathbf{x}_i | \mathbf{t}, \boldsymbol{\pi} &\sim p(\mathbf{x}_i | \mathbf{t}, \boldsymbol{\pi}, \mathbf{z}), \\ \mathbf{z} &\sim p(\mathbf{z}), \\ \mathbf{t}, \boldsymbol{\pi} &\sim \text{Coalescent}(n), \end{aligned} \tag{1} \text{eq:model}$$

$p(\mathbf{x}_i | \mathbf{t}, \boldsymbol{\pi}, \mathbf{z})$  is the data likelihood and  $p(\mathbf{z})$  is the prior distribution for the internal nodes of the tree and the pair  $\{\mathbf{t}, \boldsymbol{\pi}\}$  is provided with a prior distribution over binary tree structures known as Kingman's coalescent.

## 2.1 Kingman's coalescent

The  $n$ -coalescent is a continuous-time Markov chain originally introduced to describe the common genealogy of a sample of  $n$  individuals backwards in time (Kingman, 1982a,b). It defines a prior over binary trees with  $n$  leaves, one for each individual. The coalescent assumes a uniform prior over tree structures,  $\boldsymbol{\pi}$ , and exponential priors on the set of  $n - 1$  merging times,  $\mathbf{t}$ .

It can be thought of as a generative process on partitions of  $\{1, \dots, n\}$  as follows

- Set  $k = 1$ ,  $t_0 = 0$ ,  $\pi_0 = \{\{1\}, \dots, \{n\}\}$ .
- While  $k < n$

- Draw  $\Delta_k \sim \text{Exponential}(\lambda_k)$  where  $\lambda_k = (n - k + 1)(n - k)/2$  is a rate parameter.
- Set  $t_k = t_{k-1} + \Delta_k$ .
- Merge uniformly two sets of  $\pi_{k-1}$  into  $\pi_k$ .
- Set  $k = k + 1$ .

Because there are  $(n - k + 1)(n - k)/2$  possible merges at stage  $k$ , any particular pair in  $\pi_i$  merges with prior expected rate 1 for any  $i$ . We can compute the prior probability of a particular configuration of the pair  $\{\mathbf{t}, \boldsymbol{\pi}\}$  as

$$p(\mathbf{t}, \boldsymbol{\pi}) = \prod_{k=1}^{n-1} \exp\left(-\frac{(n-k+1)(n-k)}{2} \Delta_k\right), \quad (2) \quad \boxed{\text{eq:prior}}$$

this is, the product of merging and coalescing time probabilities. Some properties of the  $n$ -coalescent include: (i)  $\boldsymbol{\pi}$  is uniform and independent of  $\mathbf{t}$ , (ii) it is independent of the order of sets in partition  $\pi_i$  for every  $i$  and (iii) the expected value of  $t_{n-1}$  (last coalescing time) is  $\mathbb{E}[t_{n-1}] = 2(1 - n^{-1})$ .

### 2.1.1 Distribution of the latent nodes

Recall that  $z_k^* \in \pi_k$  is an internal node in a binary tree with associated latent  $d$ -dimensional vector  $\mathbf{z}_k$ . Let  $z_{c_1}^*$  and  $z_{c_2}^*$  be the children of  $z_k^*$  and define  $C = \{c_1, c_2\}$ . We designate the leaves of the tree with  $x_i^*$ . If we define  $p(\mathbf{z}_k | \mathbf{z}_C, \mathbf{t})$  to be the *transition density* between a child node,  $\mathbf{z}_C$ , and its parent,  $\mathbf{z}_k$ , then we can recursively define

$q(\mathbf{z}_k|\boldsymbol{\pi}, \mathbf{t})$  to be a distribution of  $\mathbf{z}_k$  as follows:

$$\begin{aligned} q(\mathbf{x}_i|\boldsymbol{\pi}, \mathbf{t}) &= \delta_{\mathbf{x}_i}, \\ q'(\mathbf{z}_k|\boldsymbol{\pi}, \mathbf{t}) &= \prod_{c \in C} \int p(\mathbf{z}_k|\mathbf{z}_c, \mathbf{t}) q(\mathbf{z}_c|\boldsymbol{\pi}, \mathbf{t}) d\mathbf{z}_c, \\ &= Z_k(\mathbf{X}, \boldsymbol{\pi}, \mathbf{t}) q(\mathbf{z}_k|\boldsymbol{\pi}, \mathbf{t}), \end{aligned}$$

where  $q(\mathbf{z}_k|\boldsymbol{\pi}, \mathbf{t})$  is a density (integrating to 1) and  $Z_k$  is the appropriate scaling factor.

Recently, Teh et al. (2008) showed that by using an agglomerative approach for constructing  $\{\mathbf{t}, \boldsymbol{\pi}\}$ , the likelihood for the model in equation (1) can be recursively written as

$$p(\mathbf{X}|\mathbf{t}, \boldsymbol{\pi}) = \prod_{k=1}^{n-1} Z_k(\mathbf{X}|\boldsymbol{\pi}, \mathbf{t}). \quad (3) \quad \boxed{\text{eq:lik}}$$

Because of the tree structure,  $\mathbf{z}_k$  is independent of  $\mathbf{X}$  conditional on the distributions of its two child nodes. This implies that  $Z_k(\mathbf{X}|\boldsymbol{\pi}, \mathbf{t}) = Z_k(\mathbf{X}|\pi_{1:k}, \mathbf{t}_{1:k})$ . Our formulation is equivalent to using *message passing* to marginalize recursively from the leaves to the root of the tree (Pearl, 1988). The message is  $q(\mathbf{z}_k|\boldsymbol{\pi}, \mathbf{t})$  for node  $z_k^*$  and it summarizes the entire subtree below node  $z_k^*$ .

Figure 1 illustrates the process for a segment of a tree. The size of partitions  $\pi_k$  shrink as  $k$  increases, so from the illustration  $\pi_0 = \{\{1\}, \{2\}, \dots, \{n\}\}$ ,  $\pi_1 = \{\{1, 2\}, \dots, \{n\}\}$ ,  $\pi_{k-1} = \{c_1, c_2, \dots\}$  and  $\pi_k = \{c_1 \cup c_2, \dots\}$ .

The joint distribution needed to perform inference can be obtained by combining likelihood and prior in equations (3) and (2) as

$$p(\mathbf{X}, \mathbf{t}, \boldsymbol{\pi}) = \prod_{k=1}^{n-1} \exp(-\lambda_k \Delta_k) Z_k(\mathbf{X}|\pi_{1:k}, \mathbf{t}_{1:k}). \quad (4) \quad \boxed{\text{eq:joint}}$$

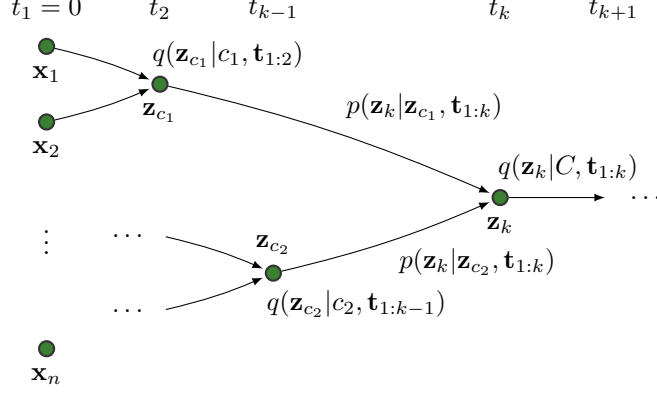


Figure 1: Binary tree structure. Latent variable  $\mathbf{t}$  and  $\boldsymbol{\pi}$  define merging points and merging sets, respectively.

(fg:stree)

## 2.2 Gaussian transition distributions

A common approach to correlated continuous data is the use of a multivariate Gaussian distribution for the transition density,

$$p(\mathbf{z}_k | \mathbf{z}_c, \mathbf{t}_{1:k}) = \mathcal{N}(\mathbf{z}_k | \mathbf{z}_c, \Delta_{k,c} \boldsymbol{\Phi}),$$

where  $\boldsymbol{\Phi}$  is a covariance matrix encoding the correlation structure in  $\mathbf{z}_k$  and  $\Delta_{k,c}$  is the time elapsed between  $t_k$  and  $t_c$ , not necessarily  $t_k - t_{k-1}$  as can be seen in Figure 1. We denote the time at which node  $z_c^*$  was created as  $t_c$ . Individual terms of the likelihood in equation (3) can be computed using

$$\begin{aligned} q(\mathbf{z}_k | \boldsymbol{\pi}_{1:k}, \mathbf{t}_{1:k}) &= \mathcal{N}(\mathbf{z}_k | \mathbf{m}_{c_1}, \tilde{s}_{c_1} \boldsymbol{\Phi}) \mathcal{N}(\mathbf{z}_k | \mathbf{m}_{c_2}, \tilde{s}_{c_2} \boldsymbol{\Phi}), \\ &= Z_k(\mathbf{X} | \boldsymbol{\pi}_{1:k}, \mathbf{t}_{1:k}) \mathcal{N}(\mathbf{z}_k | s_k(\tilde{s}_{c_1}^{-1} \mathbf{m}_{c_1} + \tilde{s}_{c_2}^{-1} \mathbf{m}_{c_2}), s_k \boldsymbol{\Phi}), \end{aligned}$$

where  $\mathbf{m}_{c_1}$  and  $s_{c_1}$  are mean and variance of  $q(\mathbf{z}_{c_1}|\boldsymbol{\pi}, \mathbf{t})$ , respectively. Furthermore,  $\tilde{s}_{c_1} = \Delta_{k,c_1} + s_{c_1}$ . This leads to  $s_k = (\tilde{s}_{c_1}^{-1} + \tilde{s}_{c_2}^{-1})^{-1}$  and the normalization constant is

$$Z_k(\mathbf{X}|\pi_{1:k}, \mathbf{t}_{1:k}) = (2\pi)^{-d/2} |v_k \boldsymbol{\Phi}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{m}_{c_1} - \mathbf{m}_{c_2}) v_k^{-1} \boldsymbol{\Phi}^{-1} (\mathbf{m}_{c_1} - \mathbf{m}_{c_2})^\top\right), \quad (5) \quad \boxed{\text{eq:ZkNorm}}$$

where  $v_k = \tilde{s}_{c_1} + \tilde{s}_{c_2} = 2\Delta_k + r_k$  and  $r_k = 2t_{k-1} - t_{c_1} - t_{c_2} + s_{c_1} + s_{c_2}$ . Note that  $\Delta_{c_1} = \Delta_{c_2}$  only if  $c_1$  and  $c_2$  are singletons. The term  $r_k$  can be interpreted as the accumulated variance up to  $t_{k-1}$  given partition  $\pi_{k-1}$ , i.e. it summarizes the two subtrees encompassed by sets  $c_1$  and  $c_2$ .

### 3 Inference

`<sc:inference>`

Inference is carried out using a sequential Monte Carlo (SMC) sampling based upon equation (4) (see Doucet et al., 2001). Define  $\mathcal{C}_i$  to be the set of all pairs of elements from the partition  $\pi_i$ , and for  $C \in \mathcal{C}_i$  define  $\pi_{i,C}$  to be the partition obtained by merging the two elements in  $C$ . For a set of  $M$  particles, we approximate the posterior of the pair  $\{\mathbf{t}, \boldsymbol{\pi}\}$  using a weighted sum of point masses obtained iteratively by drawing coalescing times  $t_k$  and chain states  $\pi_k$  one at a time from their posterior as

$$p(\Delta_k, \pi_{k-1,C} | \mathbf{t}_{1:k-1}, \pi_{1:k-1}) = Z^{-1} \exp(-\lambda_k \Delta_k) Z_k(\mathbf{X} | \pi_{1:k-1}, \pi_{k-1,C}, \mathbf{t}_{1:k-1}, \Delta_k), \quad (6) \quad \boxed{\text{eq:delta_pi_post}}$$

$$Z = \sum_{c \in \mathcal{C}_k} \underbrace{\int \exp(-\lambda_k \Delta_k) Z_k(\mathbf{X} | \pi_{1:k-1}, \pi_{k-1,c}, \mathbf{t}_{1:k-1}, \Delta_k) d\Delta_k}_{Z_{k,c}}, \quad (7) \quad \boxed{\text{eq:Zkc}}$$

From equation (7) we see that the integral needs to be computed for every pair in  $\pi_{k-1}$  at every iteration of the sampler, simply because the rate of the exponential distribution,  $\lambda_k$ , is a function of  $k$ . Algorithms introduced by Teh et al. (2008) try to



avoid the computational complexity of using equation (6) directly by simplifying it or by means of greedy alternatives. They propose for instance to draw  $\Delta_k$  from the prior; then computing  $Z_{k,C}$  is no longer necessary thus reducing computational cost. We will show that by using some properties of the distributions involved in equation (6) we can effectively decrease the computational complexity of the SMC sampler with almost no performance penalty. In particular, we will show that the most expensive parts of equations (6) and (7) need to be computed only once during inference.

If we assume a Gaussian transition distribution, we can rewrite and then expand equation (6) as

$$\begin{aligned} p(\Delta_k, \pi_{k-1,C} | \mathbf{t}_{1:k-1}, \pi_{1:k-1}) &\propto Z_k(\mathbf{X} | \pi_{k-1}, C, \mathbf{t}_{1:k}) \exp\left(-\frac{\lambda_k}{2}(2\Delta_k + r_k)\right) \exp\left(\frac{\lambda_k}{2}r_k\right), \\ &\propto \text{tGIG}_{r_k}(2\Delta_k + r_k | \boldsymbol{\epsilon}_{k-1,C}, \lambda_k) Z_{k,C}, \end{aligned} \quad (8) \quad \boxed{\text{eq:delta\_pi\_modp}}$$

where  $\boldsymbol{\epsilon}_{k-1,C} = (\mathbf{m}_{c_1} - \mathbf{m}_{c_2})\boldsymbol{\Phi}^{-1}(\mathbf{m}_{c_1} - \mathbf{m}_{c_2})^\top$  and  $\text{tGIG}_{r_k}(\lambda_k, \chi, \psi)$  is the generalized inverse Gaussian (GIG) with parameters  $\{\lambda_k, \chi, \psi\}$  truncated below  $r_k$  (Jørgensen, 1982) and the last term to the right hand side of equation (8) is used to denote that pair  $C$  is selected with probability proportional to  $Z_{k,C}$ .

$$Z_{k,C} \approx \frac{K_{1-d/2}(\sqrt{\lambda_k \boldsymbol{\epsilon}_{k-1,C}})}{(\lambda_k \boldsymbol{\epsilon}_{k-1,C}^{-1})^{(1-d/2)/2}} \exp\left(\frac{\lambda_k}{2}r_k\right), \quad (9) \quad \boxed{\text{eq:pi\_modpost}}$$

where  $K_\nu(z)$ , the modified Bessel function of second kind (Abramowitz and Stegun, 1965), is the normalizer of the generalized inverse Gaussian distribution. Since the normalization constant of the truncated GIG distribution does not have a closed form and assuming that the posterior distribution of  $\Delta_k$  is peaked, we approximate the true normalizer with  $K_\nu(z)$ . Later in the paper we explore empirically the effects of this approximation with some artificially generated data. The details on how to obtain

equations (8) and (9) can be found in Appendix A.

From Equation (8) we have

$$\Delta_k | C, \mathbf{t}_{1:k-1} \sim \text{tGIG}_{r_k}(2\Delta_k + r_k | \boldsymbol{\epsilon}_{k,C}, \lambda_k), \quad (10) \quad \boxed{\text{eq:delta\_sample}}$$

$$\pi_k | \pi_{k-1}, \mathbf{t}_{1:k-1} \sim \text{Discrete}(\pi_{k-1,C} | \mathbf{w}_{k-1}), \quad (11) \quad \boxed{\text{eq:pi\_sample}}$$

where  $\mathbf{w}_{k-1}$  is the vector of normalized weights ranging over  $\mathcal{C}_{k-1}$ , computed using equation (9). Sampling from equations (9), (10) and (11) have useful properties, (i) the conditional posterior of  $\pi_k$  does not depend on  $\Delta_k$ . (ii) Sampling from  $\Delta_k$  amounts to drawing from a truncated generalized inverse Gaussian distribution. (iii) We do not need to sample  $\Delta_k$  for every pair in  $\pi_{k-1}$ , in fact we only need to do so for the merging pair. (iv) Although  $\lambda_k$  in equation (9) changes with  $k$ , the most expensive computation,  $\boldsymbol{\epsilon}_{k-1,C}$ , needs to be computed only once. (v) The expression in equation (9) can be seen as the core of a distribution for  $\mathbf{m}_{c_1} - \mathbf{m}_{c_2}$  which has heavier tails than a Gaussian distribution, and  $Z_{k,C} \rightarrow \infty$  as  $\boldsymbol{\epsilon}_{k-1,C} \rightarrow 0$  for  $d > 1$ . Furthermore, we can rewrite equation (9) as

$$Z_{k,C} \propto \frac{K_{d/2-1}(\sqrt{\lambda_k \boldsymbol{\epsilon}_{k-1,C}})}{(\lambda_k^{-1} \boldsymbol{\epsilon}_{k-1,C})^{(d-2)/4}} \exp\left(\frac{\lambda_k}{2} r_k\right), \quad (12) \quad \boxed{\text{eq:pi\_modpost\_re}}$$

$$Z_{k,C} \approx \boldsymbol{\epsilon}_{k-1,C}^{-(d-1)/4} \exp(-\sqrt{\lambda_k \boldsymbol{\epsilon}_{k-1,C}}) \exp\left(\frac{\lambda_k}{2} r_k\right), \quad (13) \quad \boxed{\text{eq:pi\_modpost\_ne}}$$

where in equation (12) we have made a change of variables before marginalizing out  $v_k = 2\Delta_k + r_k$ , and in equation (13) we have used the limiting form of  $K_\nu(z)$  as  $z \rightarrow \infty$  (Abramowitz and Stegun, 1965). Eltoft et al. (2006) have called equation (12) the core of their multivariate Laplace distribution,  $\mathbf{m}_{c_1} - \mathbf{m}_{c_2} \sim \text{ML}(\lambda_k, \boldsymbol{\Psi})$  with parameters  $\lambda_k$  and  $\boldsymbol{\Psi}$ . When  $d = 1$  equation (13) is no longer an approximation; it is the core of a univariate Laplace distribution with rate  $\sqrt{\lambda_k \boldsymbol{\Psi}^{-1}}$ . Additionally, equation (13) can be

particularly useful as a cheap numerically stable alternative to  $K_\nu(z)$  when  $d$  is large.

### 3.1 Sampling coalescing times

Sampling from a generalized inverse Gaussian distribution is traditionally done using the *ratio-of-uniforms* method of Dagpunar (1989). We observed empirically that a slice sampler within the interval  $(r_k/2, \Delta_0 r_k/2)$  is considerably faster than the commonly used algorithm. Although we use  $\Delta_0 = 10^2$  in all our experiments, we did try larger values without noticing significant changes in the results. The slice sampler used here is a standard implementation of the algorithm described by Neal (2003b). We acknowledge that adaptively selecting  $\Delta_0$  at each step could improve the efficiency of the sampler however we did not investigate it.

### 3.2 Covariance matrix

Until now we assumed the covariance matrix  $\Phi$  as known, in most cases however the correlation structure of the observed data is unavailable. For unknown  $\Phi$  we alternate between SMC sampling for the tree structure and drawing  $\Phi$  from some suitable distribution. We do this by repeating the procedure for a number of iterations ( $N_{\text{iter}}$ ) and then dropping a subset of these as burn-in period. Because averaging of tree structures is not a well defined operation, after summarizing the posterior samples of the hyperparameters of the covariance function using medians, we perform a final SMC step to obtain a final tree structure.

For cases when observations exhibit additional structure, such as temporal or spatial data, we may assume the latent variable  $\mathbf{z}_k$  is drawn from a Gaussian process. We suppose that elements of  $\Phi$  are computed using  $\phi_{ij} = g(i, j, \boldsymbol{\theta})$  for a set of hyperpa-

rameters  $\boldsymbol{\theta}$ . For example, we could use a squared exponential covariance function

$$g(i, j, \ell, \sigma^2) = \exp\left(-\frac{1}{2\ell}d_{ij}^2\right) + \sigma^2\delta_{ij}, \quad (14) \quad \boxed{\text{eq:sqexp_noise}}$$

where  $\boldsymbol{\theta} = \{\ell, \sigma^2\}$ ,  $\delta_{ij} = 1$  IFF  $i = j$  and  $d_{ij}$  is the time between samples  $i$  and  $j$ . The smoothness of the process is controlled by the inverse length scale  $\ell$  and the amount of idiosyncratic noise by  $\sigma^2$ . The elements of  $\boldsymbol{\theta}$  are sampled by coordinate-wise slice sampling using the following function as proxy for the elements of  $\boldsymbol{\theta}$ ,

$$f(\boldsymbol{\theta}|\pi, \mathbf{t}) = \sum_{k=1}^{n-1} Z_k(\mathbf{X}|\pi_{1:k}, \mathbf{t}_{1:k}).$$

This approach is generally appropriate for continuous signals where smoothing of the covariance estimates is desirable. For the case when no smoothness is required but correlation structure is expected, conjugate inverse Wishart distributions for  $\boldsymbol{\Phi}$  can be considered. For i.i.d. data, a diagonal/spherical  $\boldsymbol{\Phi}$  with independent inverse gamma priors is a good choice, as already proposed by Teh et al. (2008).

### 3.3 Greedy implementation

Here we propose a method to draw a single sample with high posterior likelihood from the Coalescent. Such a sample can be built by greedily maximizing equation (4) one step at the time. This requires the computation of the mode of  $\Delta_k$  from equation (10) for every pair in  $\pi_{k-1}$  and merging the pair with smallest  $\Delta_k$ . For a given pair,  $C$  we have

$$\text{mode}(\Delta_k|C) = \frac{1}{2\lambda_k} \left( -d/2 + \sqrt{d^2/4 + \lambda_k \epsilon_{k-1,C}} \right) - \frac{1}{2}r_k$$

and the algorithm selects  $C$  such that

$$C = \underset{C'}{\operatorname{argmin}} \{ \Delta_{k,C'}, C' \in \mathcal{C}_{k-1} \}.$$

Because the posterior of  $\Delta_k$  is skewed to the right, a greedy implementation based on the mode of the distribution will on average underestimate coalescing times. If we use the posterior mean this bias can be decreased. We then propose using

$$\operatorname{mean}(\Delta_k|C) = \frac{1}{2} \underbrace{\sqrt{\frac{\epsilon_{k-1,C}}{\lambda_k}} \frac{K_{2-d/2}(\sqrt{\lambda_k \epsilon_{k-1,C}})}{K_{1-d/2}(\sqrt{\lambda_k \epsilon_{k-1,C}})}}_{\mu_{\Delta_k}} - \frac{1}{2} r_k,$$

where  $\mu_{\Delta_k}$  is the mean of  $\operatorname{GIG}(2\Delta_k + r_k | \epsilon_{k,C}, \lambda_k)$ . We use  $\mu_{\Delta_k}$  as an approximation to the mean of the truncated generalized inverse Gaussian distribution because no closed form is available. This means that although our approximation will be also biased to the left of the true coalescing time, it will be considerably less so than the proposal based on the mode of the distribution proposed by Teh et al. (2008).

We expect significant differences between the two greedy approaches only for low dimensional data sets because as  $d$  becomes large, the posterior of  $\Delta_k$  will be highly peaked thus making the distance between mean and mode too small to make the proposals distinguishable.

Computing two modified Bessel functions by brute force is more expensive and numerically unstable compared to the simple proposal based on the mode of the GIG distribution, however we can recursively compute the ratio of Bessel functions using the identity  $K_{v+1}(z) = K_{v-1} + 2vz^{-1}K_v(z)$ , (Abramowitz and Stegun, 1965) thus

$$\frac{K_{v+1}(z)}{K_v} = \frac{K_{v-1}(z)}{K_v} + \frac{2v}{z},$$

starting from the closed forms of  $K_{-0.5}$  and  $K_{0.5}$  if  $d$  is even, or rational approximations (accurate to 19 digits (Blair and Edwards, 1974)) to  $K_1$  and  $K_0$  if  $d$  is odd.

Incidentally, a similar recursion can be used to compute the variance of the GIG distribution at almost no cost since it is also a function of ratios involving  $K_{v+2}(z)$ ,  $K_{v+1}(z)$  and  $K_v(z)$ . The combination of mean and variance estimates can be used to estimate the mass of the distribution that is lost by truncating at  $r_k$ .

### 3.4 Computational cost

The computational cost of using equation (6) directly to sample from  $\mathbf{t}$  and  $\boldsymbol{\pi}$  for a single particle is  $\mathcal{O}(\kappa_1 n^3)$ , where  $\kappa_1$  is the cost of drawing the merging time of a single candidate pair (Teh et al., 2008). Using equation (8) costs  $\mathcal{O}(\kappa_2 n^3 + \kappa_1 n)$ , where  $\kappa_2$  is the cost of computing  $Z_{k,C}$  for a single candidate pair. Since  $\kappa_2 \ll \kappa_1$ , using equation (8) is much faster than previous approaches, at least for moderately large  $n$ . From a closer look at equation (9) we see that the only variables changing with  $k$  are  $\lambda_k$  and  $r_k$ . If we cache  $\boldsymbol{\epsilon}_{:,C}$ , the only costly operation in it is the modified Bessel function. We can approximate equation (9) by

$$Z_{k,C} \propto \frac{K_{1-d/2}(\sqrt{\boldsymbol{\epsilon}_{k-1,C}})}{(\boldsymbol{\epsilon}_{k-1,C}^{-1})^{(1-d/2)/2}} \exp\left(\frac{\lambda_k}{2} r_k\right), \quad (15) \quad \boxed{\text{eq:pi_fmodpost}}$$

where we have dropped  $\lambda_k$  from the Bessel function and the divisor in equation (9). This is acceptable because (i)  $K_\nu(z)$  is strictly decreasing for fixed  $\nu$  and (ii) the  $\lambda_k$  term appearing in the divisor is a constant in  $\log Z_{k,C}$ . Note that a similar reasoning can be applied to equation (13), which is cheaper and numerically more stable. Since equation (15) depends on  $k$  only through  $\lambda_k r_k$ , we have decreased the cost from  $\mathcal{O}(\kappa_2 n^3 + \kappa_3 n)$  to  $\mathcal{O}(\kappa_2 n^2 + \kappa_3 n)$ , that is, we need to compute equation (15) for every possible pair only once before selecting the merging pair at stage  $k$ , then we add  $\lambda_k r_k / 2$

(in log-domain) before sampling its merging time. From now on we use MPOST1 to refer to the algorithm using equation (9) and MPOST2 to the fast approximation in equation (15).

Recently, Görür and Teh (2009) proposed an efficient SMC sampler (SMC1) for hierarchical clustering with coalesecents that although was introduced for discrete data can be easily adapted to continuous data. Their approach is based on a regenerative race process in which every possible pair candidate proposes a merging time only once leading to  $\mathcal{O}(\kappa_1 n^2)$  computational time. In principle, SMC1 has quadratic cost as does MPOST2, however since  $\kappa_2 \ll \kappa_1$  our approximation is considerably faster. In addition, we have observed empirically that at least for  $n$  in the lower hundreds, MPOST1 is also faster than SMC1.

The key difference between our approach and that of Görür and Teh (2009) is that the latter proposes merging times for every possible pair and selects the pair to merge as the minimum available at a given stage whereas our approach selects the pair to merge and samples the merging time in a separate step. Additionally, they do not sample merging times using  $\Delta_{k,c} = t_k - t_c$  but directly  $t_k|t_c$ , where  $t_c$  is the time at which the pair  $c$  was created, thus  $0 \leq t_c < t_k$ . For example all pairs of singletons get created at  $t_c = 0$  regardless of the value of  $k$ . As  $t_c$  occurs generally earlier than  $t_{k-1}$ , SMC1 draws  $\Delta_{k,c}$  in larger jumps compared to MPOST1/2. Since the conditioning of  $t_k$  is involved in the truncation level of the GIG distribution, time samples from SMC1 are less constrained. This suggests that our approach will have in general better mixing properties; we compare the speed and accuracy of our approach with previous approaches on both simulated and real data examples in Appendix C.

## 4 Application: Multiple correlated outcomes

`<sec:emrResults>`

The broad adoption of electronic medical records has created the possibility of building predictive models for patient populations at almost any health system. One of the challenges faced when building these predictive models is identifying the appropriate outcome. This is because patients with chronic diseases are typically at risk for many different bad outcomes - often with varying levels of relatedness. In this section we will demonstrate the incorporation of the Kingman coalescent in a larger model that allows one to borrow strength across multiple outcomes with unknown levels of relatedness.

Suppose we have  $Y$ , an  $N \times P$  dimensional matrix with elements  $y_{ij}$  indicating the presence or absence of outcome  $j$  for patient  $i$ . Also, let  $X$  be a  $N \times K$  dimensional matrix of independent variables. We assume a probit regression model for the outcomes. For  $x_i$ , the  $i^{th}$  row of  $X$  we have:

$$P(y_{ij}) = \Psi(\mu_j + x_i \beta_j)$$

where the outcome specific intercept  $\mu_j$  allows joint modeling of outcomes with different rates,  $\beta_j$  is the  $K$  dimensional vector of regression coefficients for outcome  $j$  and  $\Psi$  is the cumulative distribution function for the standard normal distribution. Because all outcomes are at increased risk from the same chronic disease, we suspect that probabilities will be correlated. In order to capture this, we use the coalescent to impose a prior distribution that encourages correlation in the regression coefficients without *a priori* knowledge of the strength of relatedness or the values of the regression coefficients. Suppose  $\beta_k$  is the parent node for  $\beta_j$  and recall that  $t_k$  is merging time for



node  $k$ .

$$\begin{aligned}\beta_j|\beta_k, \mathbf{t}, \boldsymbol{\pi} &= N(\beta_k, (t_k - t_j)\Phi) \\ \mathbf{t}, \boldsymbol{\pi} &\sim \text{Coalescent}(K)\end{aligned}$$

We use an inverse Wishart prior on the covariance matrix  $\Phi$ .

We will test the model in a factor regression context examining models that predict outcomes for  $\approx 19\text{K}$  diabetic patients using data from an electronic medical record. Diabetic patients are susceptible to many different types of comorbidities - we develop predictive models for the 21 that are listed in Table 2. The presence/absence of these comorbidities was ascertained through ICD9 codes; the codes used to identify each are available in supplementary material.

We conducted our experiment on data from a large regional health system - Duke University IRB protocol number PRO00060340. The patient pool consisted of all patients with a home address in a particular county and all records collected between 2007 and 2011 (inclusive).

*Construction of independent variables and outcomes.* For each patient, the data consists of a collection of date-time stamped observations with labels. The labels include medical codes (ICD9, CPT), vitals, medications and laboratory values. For some types of observations (labs and vitals) there are additional continuous values, but these were discarded for this analysis. A high-dimensional sparse vector (3865 independent variables) was constructed for each patient by counting observations for 6 months prior to a specified *threshold date* - 1/1/2009 for training and 1/1/2010 for test. We used non-negative matrix factorization (Lee and Seung, 1999) on the square root of this matrix to reduce the dimension from 3865 to 25; the 25 factor scores vectors were then used in the regression model above.

We recognize the opportunity to incorporate the dimension reduction into a larger model rather than using this two-step process of dimension reduction followed by regression. However, that approach would have complicated the comparison to more standard regression techniques. We also note that the matrix factorization generates collections of observation types with very nice medical interpretations; for example, one factor is heavily weighted with medical codes, meds and labs associated with lung cancer. However, examination of those discovered relationships is outside the scope of this article.

Outcome	Training data set, 10-fold cross-validation					Validation data set				
	# Events	CR	RF	MLE	Lasso	# Events	CR	RF	MLE	Lasso
Death	450	<b>0.814</b>	0.814	0.81	0.791	376	<b>0.789</b>	0.784	0.786	0.761
Acute MI	190	<b>0.704</b>	0.658	0.7	0.657	197	<b>0.699</b>	0.648	0.699	0.694
Amputation	42	<b>0.75</b>	0.684	0.709	0.443	38	<b>0.758</b>	0.605	0.705	0.504
Aneurysm	25	0.676	<b>0.737</b>	0.64	0.355	16	<b>0.735</b>	0.582	0.706	0.508
Angioplasty	72	<b>0.616</b>	0.59	0.575	0.39	71	<b>0.605</b>	0.544	0.578	0.543
Arterial Cath	82	<b>0.759</b>	0.724	0.737	0.476	101	<b>0.673</b>	0.673	0.653	0.543
Atrial Fibrillation	619	0.817	<b>0.828</b>	0.818	0.809	593	0.799	<b>0.816</b>	0.798	0.798
Bipolar	199	<b>0.702</b>	0.649	0.694	0.603	216	<b>0.717</b>	0.691	0.711	0.548
Cardiac Cath	252	<b>0.648</b>	0.605	0.644	0.477	258	<b>0.668</b>	0.619	0.653	0.545
Skin Ulcer	286	0.727	0.692	<b>0.733</b>	0.655	338	0.705	0.689	<b>0.711</b>	0.673
COPD	296	0.726	<b>0.729</b>	0.728	0.704	294	<b>0.721</b>	0.7	0.718	0.713
Coronary Disease	1439	0.752	<b>0.779</b>	0.752	0.743	1406	0.747	<b>0.761</b>	0.747	0.732
Depression	194	0.715	0.693	<b>0.718</b>	0.707	176	0.733	0.71	0.737	<b>0.748</b>
Heart Failure	470	<b>0.806</b>	0.801	0.806	0.799	556	<b>0.785</b>	0.756	0.785	0.774
Kidney Disease	456	0.781	<b>0.807</b>	0.776	0.768	415	0.765	<b>0.787</b>	0.767	0.748
Neurological	636	<b>0.751</b>	0.745	0.75	0.745	668	0.734	0.717	<b>0.736</b>	0.73
Obesity	1647	0.676	0.675	<b>0.677</b>	0.664	1926	<b>0.639</b>	0.639	0.639	0.625
Ophthalmic	491	0.739	<b>0.744</b>	0.738	0.736	480	0.737	<b>0.741</b>	0.739	0.724
Arthritis	57	<b>0.616</b>	0.486	0.601	0.362	62	0.602	0.598	<b>0.604</b>	0.431
Stroke	344	<b>0.741</b>	0.698	0.737	0.727	334	<b>0.692</b>	0.67	0.687	0.68
Unstable Angina	173	<b>0.66</b>	0.616	0.657	0.501	125	0.72	0.689	0.712	<b>0.729</b>

Table 2: Comparison of predictive performance of different algorithms. CR=coalescent regression, RF=random forest, MLE=maximum likelihood. We also compared elastic net and ridge; because those approaches showed worse performance in all outcomes we omitted them from this table for clarity. The full table is available in supplementary material.

<tbl:results>

Outcomes were constructed by identifying the presence/absence of the relevant comorbidity in the year following the threshold date. Training and test data sets were constructed by thresholding the dataset at Jan 1, 2009 and Jan 1, 2010 respectively. We compare our results to ridge regression, elastic net and maximum likelihood regression.

*Predictive accuracy.* Predictive accuracy was assessed by area under the ROC curve using 10-fold cross-validation in the training set and directly on the test set after training on the full training set. Table 2 shows accuracies for each of the outcomes and a number of the modeling methodologies. The model described here, coalescent regression, consistently outperformed other approaches. Elastic net and Ridge were also tested, but performed worse than the others listed. The full table including elastic net and ridge is available in supplemental material.

*Future costs for diabetic patients.* Identifying high risk patients with the goal of preventing morbidity and mortality is of paramount interest. However, improving the health of the high risk patients can have the additional side effect of lowering their future healthcare costs. In order to assess the future costs of patients with diabetes, we linked the billing codes from the medical records data set with the publicly available physician fee schedule from the Centers for Medicare and Medicaid (CMS, 2015). This allows us to assign a cost to each procedure performed for each patient. We then computed the cumulative cost at each day following the threshold date for each patient.

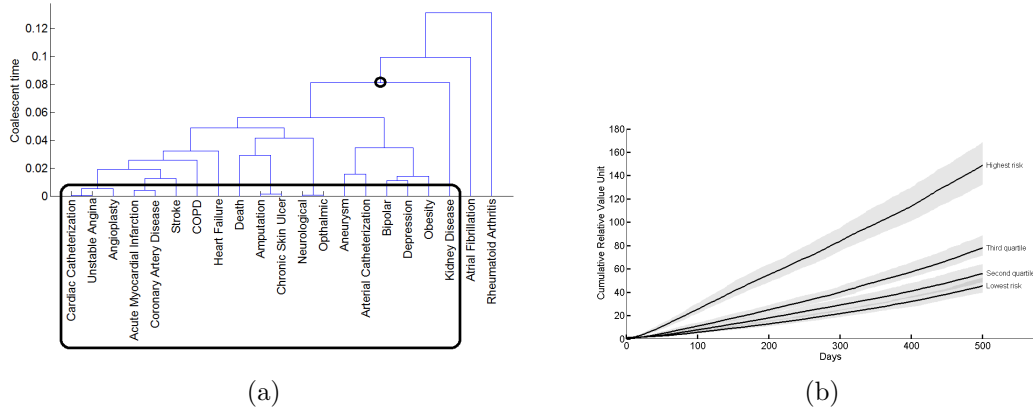


Figure 2: Future costs for patients with diabetes. Panel (a) shows the tree structure with highest posterior probability; the circled node corresponds to the model that best stratifies patients on future costs in the *training* data set. Panel (b) shows future costs of patients in the *test* data set stratified by quartile from the risk model identified in panel (a). 90% confidence bands are computed by bootstrapping with 500 patients.

`<fig:futureCost>`

One advantage of the tree model for predicting outcomes is the availability of regression models at each of the parent nodes. Models in these nodes can be loosely interpreted as predictors of the composite outcome composed of outcomes in each of their associated leaf nodes. Each of the leaf and internal node models gives rise to a separate risk stratification. We tested - in the training data set - the ability of all models to stratify patients based on future costs. Figure 2(a) shows the tree with the highest posterior probability among those sampled; the circled node corresponds to the model that best stratifies patients based on future cost in the *training* data set. Costs for each of four quartiles based on risk stratification using this model are shown in the *test* data set in Figure 2(b). Because the outcomes we are predicting with our risk model are typically high cost outcomes, we find the resulting models stratify future costs very well even though future cost was not explicitly included in the model.

## Supplementary materials

**Source code:** Matlab/C code required to perform the methods described in the manuscript, including sample scripts and the artificial data generator used throughout the results section (mvbtree\_v0.1.zip, compressed file).

## A Tree posteriors details

delta\_pi\_details)

Recall that  $v_k = 2\Delta_k + r_k$ ,  $\lambda_k = (n-k+1)(n-k)/2$ , and  $\epsilon_{k-1,C} = (\mathbf{m}_{c_1} - \mathbf{m}_{c_2})\Phi^{-1}(\mathbf{m}_{c_1} - \mathbf{m}_{c_2})^\top$ . We can combine equations (5) and (6) to get:

$$\begin{aligned} p(\Delta_k, \pi_{k-1,C} | \pi_{1:k-1}, \mathbf{t}_{1:k-1}) &= \mathcal{N}(\mathbf{m}_{c_1} - \mathbf{m}_{c_2} | \mathbf{0}, v_k \Phi) \exp(-v_k | \lambda_k / 2) \exp(r_k \lambda_k / 2), \\ &= \frac{\lambda_k}{2} (2\pi)^{-d/2} |\Phi|^{-1/2} \underbrace{|v_k|^{-d/2} \exp\left(-\frac{\epsilon_{k-1,C}}{2} v_k^{-1} - \frac{\lambda_k}{2} v_k\right)}_{\text{eq:gig_core}} \exp\left(\frac{\lambda_k}{2} r_k\right), \end{aligned} \quad (16)$$

We recognize the segment in braces of equation (16) as the core of a generalized inverse Gaussian distribution (Jørgensen, 1982) defined as

$$\text{GIG}(x | \lambda_k, \chi, \psi) = \frac{(\psi \chi^{-1})^{(1-d/2)/2}}{2K_{1-d/2}(\sqrt{\psi \chi})} x^{\lambda_k-1} \exp\left(-\frac{\chi}{2} x^{-1} - \frac{\psi}{2} x\right),$$

where  $K_\nu(z)$  is the modified Bessel function of second kind with parameter  $\nu$ .

We can thus use equation (16) to obtain

$$\begin{aligned} \Delta_k | \pi_{1:k}, \mathbf{t}_{1:k-1} &\sim \text{tGIG}_{r_k}(v_k | \epsilon_{k,C}, \lambda_k), \\ \pi_k | \mathbf{t}_{1:k-1}, \pi_{1:k-1} &\propto \underbrace{\frac{\lambda_k^2}{4} (2\pi)^{-d/2} \frac{2K_{1-d/2}(\sqrt{\lambda_k \epsilon_{k-1,C}})}{(\lambda_k \epsilon_{k-1,C}^{-1})^{(1-d/2)/2}} |\Phi|^{-1/2} \exp(\frac{\lambda_k}{2} r_k)}_{Z_{k,C}}, \\ \pi_k | \mathbf{t}_{1:k-1}, \pi_{k-1} &\leq h \underbrace{\frac{K_{1-d/2}(\sqrt{\lambda_k \epsilon_{k-1,C}})}{(\lambda_k \epsilon_{k-1,C}^{-1})^{(1-d/2)/2}} |\Phi|^{-1/2} \exp(\frac{\lambda_k}{2} r_k)}_{Z_{k,C}}, \end{aligned}$$

where the distribution for  $\Delta_k$  is truncated below  $r_k$  because of the constraint  $v_k = 2\Delta_k + r_k$ , with  $r_k > 0$ ,  $h$  is a constant term and the inequality is the result of not being able to obtain a closed form for the normalization constant of the truncated generalized inverse Gaussian distribution.

## B Full results table from Section 4

Outcome	# Events	Training data set, 10-fold cross-validation					Ridge	# Events	Validation data set					
		CR	RF	MLE	Lasso	Enet			CR	RF	MLE	Lasso	Enet	Ridge
Death	450	<b>0.814</b>	0.814	0.81	0.791	0.79	0.794	376	<b>0.789</b>	0.784	0.786	0.761	0.761	0.762
Acute MI	190	<b>0.704</b>	0.658	0.7	0.657	0.673	0.676	197	<b>0.699</b>	0.648	0.699	0.694	0.539	0.697
Amputation	42	<b>0.75</b>	0.684	0.709	0.443	0.356	0.445	38	<b>0.758</b>	0.605	0.705	0.504	0.504	0.504
Aneurysm	25	0.676	<b>0.737</b>	0.64	0.355	0.335	0.382	16	<b>0.735</b>	0.582	0.706	0.508	0.508	0.508
Angioplasty	72	<b>0.616</b>	0.59	0.575	0.39	0.415	0.406	71	<b>0.605</b>	0.544	0.578	0.543	0.543	0.543
Arterial Cath	82	<b>0.759</b>	0.724	0.737	0.476	0.517	0.568	101	<b>0.673</b>	0.673	0.653	0.543	0.543	0.543
Atrial Fibrillation	619	0.817	<b>0.828</b>	0.818	0.809	0.814	0.814	593	0.799	<b>0.816</b>	0.798	0.798	0.793	0.793
Bipolar	199	<b>0.702</b>	0.649	0.694	0.603	0.579	0.617	216	<b>0.717</b>	0.691	0.711	0.548	0.668	0.704
Cardiac Cath	252	<b>0.648</b>	0.605	0.644	0.477	0.504	0.503	258	<b>0.668</b>	0.619	0.653	0.545	0.545	0.545
Skin Ulcer	286	0.727	0.692	<b>0.733</b>	0.655	0.629	0.66	338	0.705	0.689	<b>0.711</b>	0.673	0.659	0.664
COPD	296	0.726	<b>0.729</b>	0.728	0.704	0.689	0.693	294	<b>0.721</b>	0.7	0.718	0.713	0.701	0.712
Coronary Disease	1439	0.752	<b>0.779</b>	0.752	0.743	0.743	0.743	1406	0.747	<b>0.761</b>	0.747	0.732	0.732	0.736
Depression	194	0.715	0.693	<b>0.718</b>	0.707	0.67	0.709	176	0.733	0.71	0.737	<b>0.748</b>	0.748	0.746
Heart Failure	470	<b>0.806</b>	0.801	0.806	0.799	0.798	0.8	556	<b>0.785</b>	0.756	0.785	0.774	0.773	0.773
Kidney Disease	456	0.781	<b>0.807</b>	0.776	0.768	0.768	0.768	415	0.765	<b>0.787</b>	0.767	0.748	0.743	0.746
Neurological	636	<b>0.751</b>	0.745	0.75	0.745	0.746	0.747	668	0.734	0.717	<b>0.736</b>	0.73	0.73	0.731
Obesity	1647	0.676	0.675	<b>0.677</b>	0.664	0.666	0.669	1926	<b>0.639</b>	0.639	0.639	0.625	0.629	0.63
Ophthalmic	491	0.739	<b>0.744</b>	0.738	0.736	0.734	0.733	480	0.737	<b>0.741</b>	0.739	0.724	0.724	0.724
Arthritis	57	<b>0.616</b>	0.486	0.601	0.362	0.405	0.394	62	0.602	0.598	<b>0.604</b>	0.431	0.431	0.431
Stroke	344	<b>0.741</b>	0.698	0.737	0.727	0.725	0.725	334	<b>0.692</b>	0.67	0.687	0.68	0.68	0.683
Unstable Angina	173	<b>0.66</b>	0.616	0.657	0.501	0.484	0.561	125	0.72	0.689	0.712	<b>0.729</b>	0.584	0.726

## C Numerical results

`<sc:results>`

In this section we consider a number of experiments on both artificial and real data to highlight the benefits of the proposed approaches as well as to compare them with previously proposed ones. In total three artificial data based simulations and two real data set applications are presented. All experiments are obtained using a desktop machine with 2.8GHz processor with 8GB RAM and run times are measured as single core CPU times. All methods but standard hierarchical clustering were coded by the authors using Matlab and C in order to make run times a fair proxy measurement to computational cost.

### C.1 Artificial data - structure

First we compare different sampling algorithms on artificially generated data using  $n$ -coalecscents and Gaussian processes with known squared exponential covariance functions as priors. We generated 50 replicates of two different settings,  $D_1$  and  $D_2$  of sizes  $\{n, d\} = \{32, 32\}$  and  $\{64, 64\}$ , respectively. We compare four different algorithms, POST-POST (Teh et al., 2008), SCM1 (Görür and Teh, 2009), MPOST1 and MPOST2. In each case we collect  $M = 100$  particles and set the covariance function parameters  $\{\ell, \sigma^2\}$  to their true values. As performance measures we track runtime (RT) as proxy to the computational cost, mean squared error (MSE), mean absolute error (MAE) and maximum absolute bias (MAB) of  $\log(\mathbf{t})$  and  $\boldsymbol{\pi}$ .

For  $\boldsymbol{\pi}$  we compute the distance matrix encoded by  $\{\mathbf{t}, \boldsymbol{\pi}\}$ . In addition, we compute the difference between true and estimated last coalescing times (TD) as a way to quantify estimation bias, thus positive and negative differences imply under and over estimation, respectively.

Table 4 shows performance measures averaged over 50 replicates for each data set.

Set	Measure	POST-POST	MPost1	SMC1	MPost2
Merge time (t)					
	$10^1 \times \text{MSE}$	$0.52 \pm 0.22$	<b><math>0.44 \pm 0.18</math></b>	$0.66 \pm 0.28$	$0.45 \pm 0.18$
$D_1$	$10^1 \times \text{MAE}$	$1.88 \pm 0.45$	<b><math>1.68 \pm 0.39</math></b>	$2.01 \pm 0.46$	$1.72 \pm 0.40$
	$10^1 \times \text{MAB}$	$5.13 \pm 1.23$	$4.93 \pm 1.12$	$6.09 \pm 1.30$	<b><math>4.91 \pm 1.08</math></b>
	$10^2 \times \text{MSE}$	$4.06 \pm 1.08$	<b><math>3.04 \pm 0.90</math></b>	$5.37 \pm 2.23$	$3.30 \pm 0.94$
$D_2$	$10^1 \times \text{MAE}$	$1.73 \pm 0.26$	<b><math>1.42 \pm 0.26</math></b>	$1.83 \pm 0.39$	$1.49 \pm 0.27$
	$10^1 \times \text{MAB}$	$4.47 \pm 0.73$	$4.40 \pm 0.78$	$5.97 \pm 1.39$	<b><math>4.39 \pm 0.73</math></b>
Last coalescing time ( $t_{n-1}$ )					
$D_1$	$10^0 \times \text{TD}$	<b><math>-0.04 \pm 0.52</math></b>	$-0.05 \pm 0.52$	$0.11 \pm 0.55$	$-0.07 \pm 0.51$
$D_2$	$10^0 \times \text{TD}$	<b><math>-0.12 \pm 0.51</math></b>	$-0.13 \pm 0.50$	$0.13 \pm 0.49$	$-0.13 \pm 0.63$
Distance matrix ( $\pi$ )					
	$10^1 \times \text{MSE}$	$0.79 \pm 0.50$	$0.70 \pm 0.49$	$1.32 \pm 0.63$	<b><math>0.70 \pm 0.42</math></b>
$D_1$	$10^1 \times \text{MAE}$	$2.24 \pm 0.78$	<b><math>2.13 \pm 0.76</math></b>	$3.03 \pm 0.90$	$2.14 \pm 0.72$
	$10^1 \times \text{MAB}$	$6.77 \pm 1.20$	$6.50 \pm 1.12$	$8.77 \pm 1.96$	<b><math>6.48 \pm 1.13</math></b>
	$10^2 \times \text{MSE}$	$5.79 \pm 3.15$	<b><math>4.89 \pm 2.92</math></b>	$11.36 \pm 5.68$	$5.27 \pm 2.94$
$D_2$	$10^1 \times \text{MAE}$	$1.95 \pm 0.68$	<b><math>1.78 \pm 0.65</math></b>	$2.85 \pm 0.83$	$1.85 \pm 0.65$
	$10^1 \times \text{MAB}$	$6.06 \pm 0.79$	<b><math>5.75 \pm 0.73</math></b>	$8.54 \pm 2.01$	$5.81 \pm 0.69$
Computational cost					
$D_1$	$10^0 \times \text{RT}$	$18.65 \pm 0.24$	$2.29 \pm 0.04$	$3.76 \pm 0.07$	<b><math>1.98 \pm 0.03</math></b>
$D_2$	$10^{-1} \times \text{RT}$	$14.50 \pm 0.06$	$1.08 \pm 0.00$	$1.39 \pm 0.01$	<b><math>0.61 \pm 0.00</math></b>

Table 4: Performance measures for structure estimation. MSE, MAE, MAB, RT and TD are mean squared error, mean absolute error, maximum absolute bias, runtime in seconds and last coalescing time difference, respectively. Figures are means and standard deviations across 50 replicates. Best results are in boldface letters.

In terms of error, we see that all four algorithms perform about the same as one might expect, however with MPOST1 and SMC1 being slightly better and slightly worse, respectively. The computational cost is significantly higher for the POST-POST approach whereas MPOST2 is the fastest. We see MPOST1 and MPOST2 consistently outperforming the other two algorithms as an indication of better mixing properties. In more general terms, MPOST2 provides the best error/computational cost trade-off as the difference in accuracy between MPOST1 and MPOST2 is rather minimal. In terms of coalescing time estimation bias, we see that POST-POST performs slightly better and SMC1 has the largest bias from all four. Note that POST-POST is not affected by the approximation introduced in equation (9) hence it should be in principle unbiased.



## C.2 Artificial data - covariance

Next we want to test the different sampling algorithms when the parameters of the Gaussian process covariance function,  $\{\ell, \sigma^2\}$ , need to be learned as well. We use settings similar to those in the previous experiment with the difference that now  $M = 50$  particles are collected and  $N_{\text{iter}} = 50$  iterations are performed to learn the covariance matrix parameters. We dropped the first 10 iterations as burn-in period. In addition to the previously mentioned performance measures we also compute MSE, MAE and MAB for the inverse length scale  $\ell$  from equation (14). In order to simplify the experiment we set a priori  $\sigma^2 = 1 \times 10^{-9}$  to match a *noiseless* scenario, however similar results are obtained when learning both parameters at the same time (results not shown). Table 5 shows an overall similar trend when compared to Table 4. In terms of covariance function parameter estimation, we see all algorithms perform about the same which is not surprising considering they use the same sampling strategy. Time difference measures, TD, indicate that all approaches underestimate the last coalescing time, although slightly less for our two proposals. More specifically, median differences between SMC1 and MPOST1/2 are significant at the 0.05 level only for the distance matrix measure, which indicates that our algorithms are better at capturing the true hierarchical structure of the data but all methods do about the same in terms of coalescing times and inverse length-scale estimation. MPOST1/2 is approximately twice as fast as SMC1. Other than computational speed, differences between MPOST1 and MPOST2 are not significant.

## C.3 Artificial data - greedy algorithm

As final simulation based on artificial data we want to test whether there is a difference in performance between the mean based greedy approach (MGREEDY) and the algorithm proposed by Teh et al. (2008) that utilizes modes (GREEDY). We gener-

Set	Measure	MPost1	SMC1	MPost2
Merge time ( $t$ )				
$D_1$	$10^1 \times \text{MSE}$	<b>1.20 <math>\pm</math> 0.51</b>	1.26 $\pm$ 0.44	1.24 $\pm$ 0.53
	$10^1 \times \text{MAE}$	3.02 $\pm$ 0.80	<b>2.98 <math>\pm</math> 0.62</b>	3.06 $\pm$ 0.81
	$10^1 \times \text{MAB}$	<b>6.43 <math>\pm</math> 1.26</b>	7.01 $\pm$ 1.50	6.52 $\pm$ 1.38
$D_2$	$10^2 \times \text{MSE}$	<b>5.38 <math>\pm</math> 1.66</b>	6.32 $\pm$ 1.94	5.61 $\pm$ 1.76
	$10^1 \times \text{MAE}$	2.02 $\pm$ 0.36	<b>2.01 <math>\pm</math> 0.36</b>	2.07 $\pm$ 0.36
	$10^1 \times \text{MAB}$	4.75 $\pm$ 0.72	6.16 $\pm$ 1.35	<b>4.73 <math>\pm</math> 0.62</b>
Last coalescing time ( $t_{n-1}$ )				
$D_1$	$10^0 \times \text{TD}$	<b>0.21 <math>\pm</math> 0.52</b>	0.43 $\pm$ 0.51	0.25 $\pm$ 0.55
$D_2$	$10^0 \times \text{TD}$	<b>0.04 <math>\pm</math> 0.44</b>	0.29 $\pm$ 0.54	0.08 $\pm$ 0.35
Distance matrix ( $\pi$ )				
$D_1$	$10^1 \times \text{MSE}$	<b>1.31 <math>\pm</math> 0.87</b>	2.85 $\pm$ 1.76	1.33 $\pm$ 0.86
	$10^1 \times \text{MAE}$	<b>2.94 <math>\pm</math> 1.19</b>	4.53 $\pm$ 1.69	2.96 $\pm$ 1.19
	$10^1 \times \text{MAB}$	<b>8.77 <math>\pm</math> 2.18</b>	9.60 $\pm$ 1.73	8.84 $\pm$ 2.13
$D_2$	$10^1 \times \text{MSE}$	<b>0.64 <math>\pm</math> 0.35</b>	1.54 $\pm$ 0.67	0.66 $\pm$ 0.34
	$10^1 \times \text{MAE}$	<b>2.08 <math>\pm</math> 0.63</b>	3.29 $\pm$ 0.94	2.08 $\pm$ 0.65
	$10^1 \times \text{MAB}$	6.77 $\pm$ 1.13	8.41 $\pm$ 1.42	<b>6.76 <math>\pm</math> 1.15</b>
Inverse length scale ( $\ell$ )				
$D_1$	$10^4 \times \text{MSE}$	<b>2.36 <math>\pm</math> 3.20</b>	2.93 $\pm$ 4.51	2.38 $\pm$ 3.23
	$10^2 \times \text{MAE}$	<b>1.17 <math>\pm</math> 0.99</b>	1.24 $\pm$ 1.09	1.18 $\pm$ 0.99
	$10^2 \times \text{MAB}$	1.40 $\pm$ 1.14	2.06 $\pm$ 2.18	<b>1.38 <math>\pm</math> 1.15</b>
$D_2$	$10^4 \times \text{MSE}$	2.86 $\pm$ 3.22	3.55 $\pm$ 4.48	<b>2.83 <math>\pm</math> 3.17</b>
	$10^2 \times \text{MAE}$	1.35 $\pm$ 1.02	1.44 $\pm$ 1.14	<b>1.34 <math>\pm</math> 1.01</b>
	$10^2 \times \text{MAB}$	1.57 $\pm$ 1.29	2.39 $\pm$ 2.24	<b>1.55 <math>\pm</math> 1.27</b>
Computational cost				
$D_1$	$10^{-1} \times \text{RT}$	5.69 $\pm$ 0.02	13.65 $\pm$ 0.06	<b>4.86 <math>\pm</math> 0.02</b>
$D_2$	$10^{-2} \times \text{RT}$	2.72 $\pm$ 0.07	5.12 $\pm$ 0.05	<b>1.49 <math>\pm</math> 0.01</b>

Table 5: Performance measures for covariance estimation. MSE, MAE, MAB, RT and DT are mean squared error, mean absolute error, maximum absolute bias runtime in seconds and last coalescing time difference, respectively. Figures are means and standard deviations across 50 replicates. Best results are in boldface letters.

cial\_covariance)

ated 50 replicates of two different settings  $D_1$  and  $D_2$  of sizes  $\{n, d\} = \{32, 32\}$  and  $\{128, 128\}$ , respectively. We run  $N_{\text{iter}}$  iterations of the algorithm and drop the first 10 samples as burn-in period. Other settings and performance measures are the same as in the previous experiment. Table 6 shows that the algorithm based on means performs consistently better than the original when the data set is small.

Although not shown, we tried other settings in between  $D_1$ ,  $D_2$  and larger than  $D_2$  with consistent results, this is, the difference between GREEDY and MGREEDY decreases with the size of the dataset. We do want to show instead how last coalescing time differences change as a function of  $d$ . For this purpose we generated 250 replicates of  $D_1$  and  $D_2$  for 6 different values of  $d$  and assumed the covariance function parameter as known. Figure 3 shows that both greedy approaches tend to underestimate the last

$D_1$			$D_2$		
Measure	GREEDY	MGREEDY	Measure	GREEDY	MGREEDY
Merge time ( $\mathbf{t}$ )					
$10^1 \times \text{MSE}$	$1.17 \pm 0.46$	<b><math>0.55 \pm 0.25</math></b>	$10^2 \times \text{MSE}$	$0.94 \pm 0.23$	<b><math>0.59 \pm 0.15</math></b>
$10^1 \times \text{MAE}$	$2.99 \pm 0.72$	<b><math>1.91 \pm 0.50</math></b>	$10^1 \times \text{MAE}$	$0.80 \pm 0.11$	<b><math>0.61 \pm 0.08</math></b>
$10^1 \times \text{MAB}$	$6.70 \pm 1.49$	<b><math>5.46 \pm 1.45</math></b>	$10^1 \times \text{MAB}$	$2.72 \pm 0.49$	<b><math>2.43 \pm 0.46</math></b>
Last coalescing time ( $t_{n-1}$ )					
$10^0 \times \text{TD}$	$0.40 \pm 0.40$	<b><math>0.24 \pm 0.39</math></b>	$10^0 \times \text{TD}$	$0.11 \pm 0.19$	<b><math>0.07 \pm 0.19</math></b>
Distance matrix ( $\pi$ )					
$10^1 \times \text{MSE}$	$1.00 \pm 0.78$	<b><math>0.59 \pm 0.50</math></b>	$10^1 \times \text{MSE}$	$0.11 \pm 0.11$	<b><math>0.08 \pm 0.08</math></b>
$10^1 \times \text{MAE}$	$2.74 \pm 1.07$	<b><math>2.04 \pm 0.88</math></b>	$10^1 \times \text{MAE}$	$0.91 \pm 0.38$	<b><math>0.76 \pm 0.33</math></b>
$10^1 \times \text{MAB}$	$8.69 \pm 1.56$	<b><math>7.26 \pm 1.49</math></b>	$10^1 \times \text{MAB}$	$3.91 \pm 0.69$	<b><math>3.59 \pm 0.66</math></b>
Inverse length scale ( $\ell$ )					
$10^4 \times \text{MSE}$	<b><math>0.29 \pm 2.94</math></b>	$0.30 \pm 2.92$	$10^4 \times \text{MSE}$	<b><math>1.37 \pm 3.54</math></b>	$1.38 \pm 3.56$
$10^2 \times \text{MAE}$	<b><math>0.54 \pm 0.96</math></b>	<b><math>0.54 \pm 0.96</math></b>	$10^2 \times \text{MAE}$	<b><math>1.17 \pm 1.03</math></b>	$1.18 \pm 1.03$
$10^2 \times \text{MAB}$	<b><math>0.62 \pm 1.07</math></b>	$0.63 \pm 1.05$	$10^2 \times \text{MAB}$	<b><math>1.23 \pm 1.06</math></b>	$1.24 \pm 1.15$
Computational cost					
$10^0 \times \text{RT}$	$2.14 \pm 0.13$	$2.16 \pm 0.16$	$10^{-1} \times \text{RT}$	<b><math>3.04 \pm 0.14</math></b>	$3.11 \pm 0.15$

Table 6: Performance measures for greedy algorithms. MSE, MAE, MAB, RT and DT are mean squared error, mean absolute error, maximum absolute bias runtime in seconds and last coalescing time difference, respectively. Figures are means and standard deviations across 50 replicates. Best results are in boldface letters.

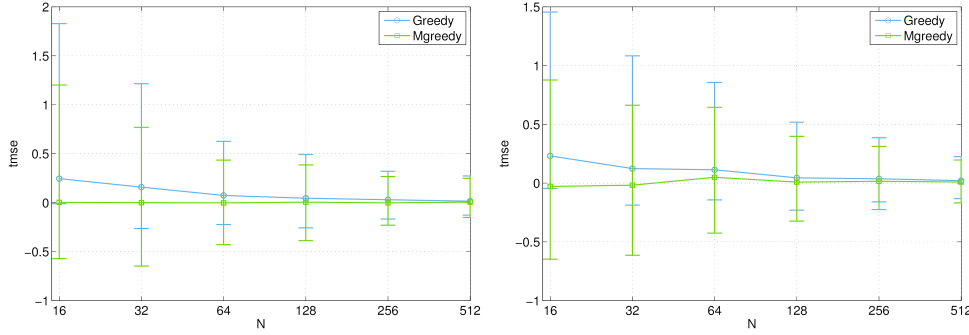


Figure 3: Last coalescing time comparison for greedy algorithms. Median time differences TD as a function of data set dimensionality  $d$ , for  $D_1$  (left) and  $D_2$  (right). Error bars represent 90% empirical quantiles.

coalescing time, however the mean based algorithm appears to be more accurate and less sensitive to the size of the dataset.

## C.4 Handwritten digits

The USPS database<sup>1</sup> contains 9289 grayscale images of digits, each  $16 \times 16$  pixels in size and scaled to fall within the range  $[-1, 1]$ . Here we use subsets of 2500 images,

<sup>1</sup>Data available from <http://cs.nyu.edu/roweis/data.html>.

50 from each digit, randomly selected from the full data set. We apply MPOST2, MGREEDY and average-link agglomerative clustering (HC) to 25 of such subsets. For the covariance matrix  $\Phi$  we use a Matérn covariance function with parameter  $\nu = 3/2$  and additive noise defined as follows

$$g(i, j, \ell_x, \ell_y, \sigma^2) = \left(1 + \frac{\sqrt{3}}{\ell_x} d_{x,ij}\right) \left(1 + \frac{\sqrt{3}}{\ell_y} d_{y,ij}\right) \exp\left(-\frac{\sqrt{3}}{\ell_x} d_{x,ij} + \frac{\sqrt{3}}{\ell_y} d_{y,ij}\right) + \sigma^2 \delta_{ij},$$

where  $d_{x,ij}$  and  $d_{y,ij}$  are distances in the two axes of the image, and we have assumed axis-wise independence (Rasmussen and Williams, 2006).

#### C.4.1 Performance metrics

As performance measures we use (i) the *subtree* score defined as  $N_{\text{subset}}/(n - K)$ , where  $N_{\text{subset}}$  is the number of internal nodes with leaves from the same class,  $n$  is the number of observations and  $K$  is the number of classes (Teh et al., 2008), and (ii) the area under the adjusted Rand index (ARI) curve (AUC), which is a similarity measure for pairs of data partitions. It takes values  $\leq 1$  where 1 indicates that the two partitions agree as much as possible given their respective sizes (Hubert and Arabie, 1985).

For the ARI metric, we do not compare the true partition and clustering directly. Given a particular clustering, we label each cluster based on voting by the members of the cluster. Subsequently, a partition is created by relabeling every observation to match the label of its cluster. This has the effect of producing perfect accuracy when the partition consists of  $n$  singleton sets. We test all  $n$  possible partitions from the clustering model and plot ARI vs number of clusters  $N_c$ . This produces a graphical representation that resembles a ROC curve; if performance is perfect, ARI will be 1 for  $N_c > K$ . Also, ARI will increase with  $N_c$ . When  $N_c = 1$  and  $N_c = n$ , ARI is always 0 and 1, respectively. Just like in a ROC curve, an algorithm is as good as its ARI's rate of change thus we can assess the overall performance by computing the area

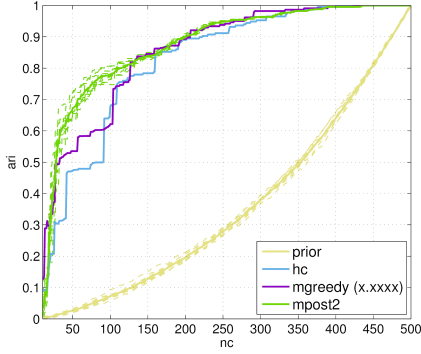


Figure 4: USPS digits results. (Left) USPS data ARI curves. PRIOR draws structures directly from a  $n$ -coalescent prior, REF is MPOST2 with diagonal covariance matrix and HC is standard hierarchical clustering with average link function and euclidean distance metric. Figures in parenthesis are AUC scores. (Right) Subtree scores, AUC is area under the ARI curve and RT is runtime in minutes. Figures are means and standard deviations across 25 replicates.

`<fig:usps_auc>`

under the ARI curve. Figure 4 shows curves for a particular data set and the three considered algorithms. We also included results obtained by tree structures drawn from the coalescent prior and MPOST2 with diagonal covariance matrix as baselines.

Table in Figure 4 shows average subset scores for the algorithms considered. We performed inference for 20 iterations and 10 particles. No substantial improvement was found by increasing  $N_{\text{iter}}$  or  $M$ . GREEDY produced the same results as MGREEDY and SMC1 did not performed better than MPOST1/2 but it took approximately 10 times longer to run (results not shown). We see that coalescent based algorithms perform better than standard hierarchical cluster in terms of AUC. MPOST1 and MPOST2 are best in subtree scores and AUC, respectively.

## C.5 Motion capture data

We apply MPOST2 to learn hierarchical structures in motion capture data (MOCAP). The data set consists of 102 time series of length 217 corresponding to the coordinates of a set of 34 three dimensional markers placed on a person breaking into run<sup>2</sup>. For

<sup>2</sup>Data available from [http://accad.osu.edu/research/mocap/mocap\\_data.htm](http://accad.osu.edu/research/mocap/mocap_data.htm).

the covariance matrix  $\Phi$ , we used the squared exponential function in equation (14). Results are obtained after running 50 iterations of MPOST2 with 50 particles. It took approximately 5 minutes to complete the run. Left panel in Figure 5 shows two subtrees containing data from all markers in the  $Y$  and  $Z$  axes.

In order to facilitate visualization, we relabeled the original markers to one of the following: head (Head), torso (Torso), right leg (Leg:R), left leg (Leg:L), right arm (Arm:R) and left arm (Arm:L). The subtrees from Figure 5 are obtained from the particle with maximum weight (0.129) at the final iteration of the run with effective sample size 24.108. We also examined the trees for the remaining particles and noted no substantial structural differences with respect to Figure 5.

The resulting tree has interesting features: (i) Sensors from the  $Y$  and  $Z$  coordinate axes form two separate clusters. (ii) Leg markers have in general larger merging times than the others, whereas the opposite is true for head markers. (iii) The obtained tree agrees with the structure of the human body reasonably well; for instance in the middle-right panel of Figure 5 we see a heat map with 9 markers, 4 of them from the head, 1 from the torso (C7, base of the neck) and 4 from the arms (shoulders and upper arms). The two arm sensors close to the torso correspond to the shoulders while the other two—with larger merging times, are located in the upper arms.

The obtained structure is fairly robust to changes in the number of iterations and particles. MPOST1, MGREEDY, SMC1, and POST-POST produce structurally similar trees to the one shown in Figure 5, however with different running times. In particular, they took 7, 1, 12 and 75 minutes, respectively.

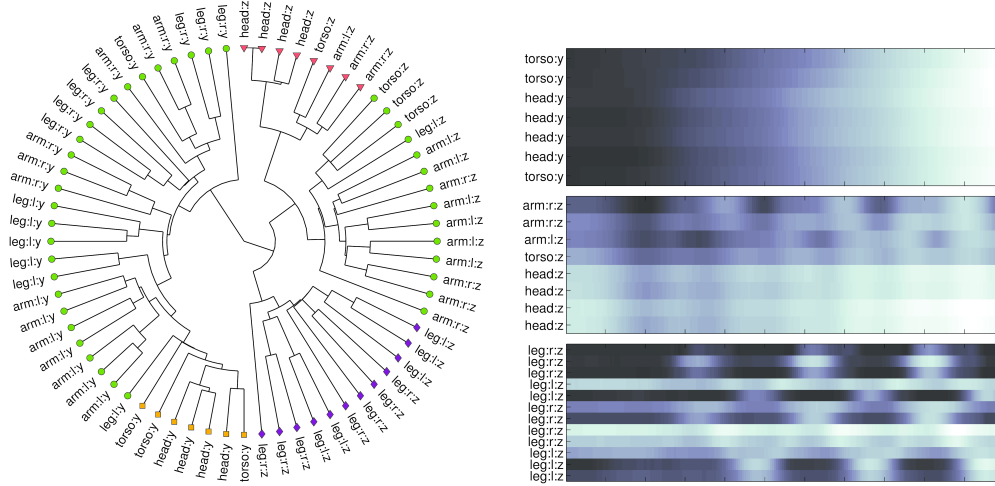


Figure 5: MOCAP data results. (Left) Resulting subtree from the particle with maximum weight (0.0784) at iteration 50. (Right) Data corresponding to three subtrees of (Left) marked with squares, triangles and diamonds, respectively.

# References

amowitz65a M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Dover Publications, New York, 1965.

adams10 R. P. Adams, Z. Ghahramani, and M. I. Jordan. Tree-structured stick breaking for hierarchical data. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 19–27. MIT Press, 2010.

blair74a J. M. Blair and C. A. Edwards. Stable rational minimax approximations to the modified Bessel functions  $i_0(x)$  and  $i_1(x)$ . Technical Report 4928, Atomic Energy of Canada Limited, Chalk River Nuclear Laboratories, 1974.

cmsFee CMS. Physician fee schedule, 2015. URL <https://www.cms.gov/Medicare/Medicare-Fee-for-Service-Payment/PhysicianFeeSched/index>

dagpunar89a J. S. Dagpunar. An easily implemented generalised inverse Gaussian generator. *Communications in Statistics - Simulation and Computation*, 18(2):703–710, 1989.

- doucet01** A. Doucet, N. de Freitas, N. Gordon, and A. Smith. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
- eltoft06a** T. Eltoft, T. Kim, and T.-W. Lee. On the multivariate Laplace distribution. *IEEE Signal Processing Letters*, 13(5):300–303, 2006.
- gorur08** D. Görür and Y. W. Teh. An efficient sequential Monte Carlo algorithm for coalescent clustering. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 521–528. MIT Press, 2009.
- heller05** K. A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304. ACM, 2005.
- henao12a** R. Henao, J. W. Thompson, M. A. Moseley, G. S. Ginsburg, L. Carin, and J. E. Lucas. Hierarchical factor modeling of proteomics data. In *Computational Advances in Bio and Medical Sciences (ICCABS), 2012 IEEE 2nd International Conference on*, 2012.
- henao13a** R. Henao, J. W. Thompson, M. A. Moseley, G. S. Ginsburg, L. Carin, and J. E. Lucas. Latent protein trees. *Annals of Applied Statistics*, to appear, 2013.
- hubert85a** L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.
- rgensen82a** Bent Jørgensen. *Statistical properties of the generalized inverse Gaussian distribution*, volume 9 of *Lecture notes in statistics*. Springer-Verlag, 1982.
- kingman82** J. F. C. Kingman. The coalescent. *Stochastic processes and their applications*, 13(3):235–248, 1982a.
- kingman82a** J. F. C. Kingman. On the genealogy of large populations. *Journal of Applied Probability*, 19:27–43, 1982b.



- nnmf** Daniel Lee and Sebastian Seung. Learning parts of objects by non-negative matrix factorization. *Nature*, 401, 1999.
- neal03** R. Neal. Density modeling and clustering using Dirichlet diffusion trees. In J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, editors, *Bayesian Statistics 7*, pages 619–629. Oxford University Press, 2003a.
- neal03a** R. M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–741, 2003b.
- pearl88** J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- rai08** P. Rai and H. Daume III. The infinite hierarchical factor regression model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1321–1328. MIT Press, 2009.
- asmussen06** C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.
- teh08** Y. W. Teh, H. Daume III, and D. Roy. Bayesian agglomerative clustering with coalecscents. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1473–1480. MIT Press, 2008.
- zhang11a** X. Zhang, D. Dunson, and L. Carin. Tree-structured infinite sparse factor model. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 785–792, 2011.